第31卷 第15期 计 算 机 工 程 2005年8月
Vol.31 № 15 Computer Engineering August 2005
·博士论文· 文章编号：1000—3428(2005)15—0022—02 文献标识码：A 中图分类号：TP 393.08

X -window 应用系统的安全问题及其解决方案

曾剑平[1]，郭东辉[1,2]
(1. 厦门大学物理系 EDA 实验室，厦门 361005；2. 厦门大学微机电研究中心，厦门 361005)

摘 要：通过介绍 X-window 应用系统的通信原理，分析 Internet 环境下 X-window 应用系统的通信安全问题，提出了一种能够保证 X-window 实际应用安全性的解决方案，并直接应用到 Web-EDA 技术平台开发中。该安全通信的解决方案主要集成了主机隐藏、通信加密、二次验证等 3 种通信安全技术，而无须对 X-window 应用软件的原程序进行修改，就能解决其在 Internet 中通信的安全问题。

关键词：X-window；Internet 安全；认证

Security Issue X-window Application System and Its Solution

ZENG Jianping[1], GUO Donghui[1, 2]
(1. EDA Lab, Dept. of Physics, Xiamen University, Xiamen 361005;
2. Micro-Electro-Mechanical Systems Research Center, Xiamen University, Xiamen 361005)

【Abstract】 The communication principle in X-window-based application is introduced, and several security issues which may be encountered when applying X-window protocol to the Internet are analyzed. A new kind of security solution that is more practical is proposed, and it is applied to the development of Web-EDA platform. The solution integrates with three kinds of technology, i.e. mainframe hidden, communication encryption and second authentication. The X-window applications need not be modified, but the security issues can be resolved very well.

【Key words】 X-window; Internet security; Authentication

X-window 是一种 Unix 操作系统环境下支持图形窗口和应用程序远程执行的应用层协议，可以利用 X-window 协议构造基于网络的 Unix 多用户窗口环境，为分布式计算环境下各种不同计算机提供了显示的公共界面。基于 X-window 的应用软件具有 B/S 软件体系结构的特征。在实际使用时，只需要在用户的计算机上安装支持 X-window 协议的容器软件，之后用户就能进入 X-window 的主机系统，并运行所需要的软件。而整个运行过程都是在这个容器软件中进行的。X-window 的一种常见的应用方式是，不同工作站都分布在一个局域网内部，不同的客户统一登录到指定 X-window 服务器完成各自的工作。这种新的应用模式在真正实用之前，需要解决若干重要的问题，比如安全问题、网络服务质量(QoS)的保证问题等，才能适应 Internet 环境的要求。

本文分析了 Internet 环境下，X-window 应用系统可能遭遇的安全问题，并给出了一种基于协议转换的新的解决方案。

1 X-window 的通信原理

X-window 应用软件是以 C/S 的工作方式为基础的，如图 1，它的内容主要是由 3 个相关的部分组成，即称为 Xclient 的客户端、服务端 Xserver 及 X 通信协议。其中 Xclient 是在 Unix 环境下的一个应用程序，但它无法在远程终端直接控制显示输出，需要通过一个终端服务程序即 Xserver 来完成。Xserver 是以服务程序形式来实现控制用户的输入显示器、键盘、鼠标等输入输出设备，它负责解释 Xclient 发送来的消息，并根据解释结果执行相应的动作，如建立窗口、窗口中绘画等。X 通信协议则规定了双方需共同遵守的约定，说明了每个命令的格式和组成。

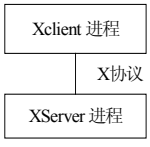

图 1 X-window 的结构

X-window 应用软件完整的一次通信过程可以分成 4 个阶段，即服务器定位与显示设备参数协商阶段、Xclient 应用与 Xserver 建立连接阶段、Xclient 应用与 Xserver 通信阶段、Xclient 应用与 Xserver 断开连接阶段。各阶段的功能如下：

(1) 服务器定位与显示设备参数协商阶段。在此阶段，支持 X 协议的用户端软件如 Xmanager、X-winpro，使用 UDP 协议与远程服务器的显示管理 XDMCP 端口（即 177 端口）上进行通信。该通信过程是由两者之间的 5 次通信握手过程组成，主要是进行显示设备的参数协商、获取主机信息、获取用户端软件标识等。

(2) Xclient 应用与 Xserver 建立连接阶段。用户端软件运行之后，即在本地启动端口监听，等待 Xclient 的连接请求。按照约定该端口采用 6000 端口，用户是不能修改配置的。如果 Xserver 接受 Xclient 的连接请求，则两者之间就在 6000 端口建立了 TCP 的 socket 连接。

(3) Xclient 应用与 Xserver 通信阶段。在此阶段，二者之间主要是传输一些与应用相关的数据。

(4) Xclient 应用与 Xserver 断开连接阶段。断开连接，释放双方的资源。

2 Internet 环境下的安全问题

从上述通信流程上可见，进行 X-window 通信的双方都需要打开通信端口，并接受对方的连接请求，并在建立连接的基础上进行应用数据通信。因此，把这种以 C/S 工作方式直接应用到 Internet 上，容易产生以下几方面的安全问题：

(1) 主机端口问题。客户端和服务器直接暴露在 Internet 上，受到针对 UDP 或 TCP 端口 177、6000 攻击的可能性增大。例如，

基金项目：国家自然科学基金 (60076015)和国家人事部留学人员创业基金联合资助项目
作者简介：曾剑平(1973－)，男，博士生，主研方向为信息安全、分布式系统；郭东辉，教授、博导
定稿日期：2004-07-09 **E-mail:** zeng_jian_ping@hotmail.com

用户可以创建一个 socket，直接与运行 Xserver 的主机的 6000 端口建立连接，并使用 X 协议与 Xserver 会话，通过 Xserver 控制和获得主机上运行的其它 X 应用程序的信息。

（2）数据传输的安全问题。X 协议是一种标准的公开协议，所以任何人只要能获得通信中的数据流，就可能分析出所传输的数据，特别是 Xclient 和 Xserver 在通信过程中的第一阶段中的参数协商过程。

（3）用户身份认证。X-window 应用系统采用单一的用户名/密码的用户身份认证方式，通过身份验证之后，用户就具有了预先分配的权限，可以运行相关的操作系统命令。这种认证方式在 Internet 上受到来自不同地点、不同时间的恶意尝试的可能性将大大增加，从而削弱了这种用户认证方式的安全性。然而，X-window 目前只提供基于 Xlib 的 C 语言图形处理开发接口[2]，没有相应的 API 的能让设计人员获得 X-window 通信过程中的数据，所以基于 X-window 的应用程序安全问题一般是在应用层解决[3]，即正确配置 X-window 系统，它是通过配置主机的允许访问或禁止访问列表来保证系统的安全。显然，这种方式在 Internet 上是很难实现的，因为用户接入 Internet 的计算机一般没有固定的 IP 地址，无法通过 IP 过滤方法来保证系统的通信安全。为了实际解决 X-window 在互联网中推广应用的安全问题，我们在开发基于 X-window 的 Web-EDA 技术平台[4]时，引入了一种以主机隐藏技术、通信加密技术、二次身份认证技术相结合的安全通信方案来保证 Web-EDA 技术平台的安全性。

3 安全通信方案的实现

本安全通信方案所采用的 3 种技术主要从链路层、网络层和应用层加强了 Internet 环境下 X-window 应用程序的安全保障。因此，对于 Internet 上的各种攻击产生的隐患都能有效地防范。下面具体说明所采用的方法。

3.1 主机隐藏

在 TCP/IP 网络中，针对主机的攻击大部分属于对 TCP 协议的攻击，主要是通过同步信号淹没、复位与结束信号攻击等方式来实现。这类攻击可以通过防火墙和 SSL 等网络安全协议来防范。而有些攻击绕过一些验证步骤直接与主机端口建立非法连接，并获得主机信息，这在 X-window 通信中就更容易发生。而防火墙或安全通信协议对此是无能为力的。

对此可以采用主机隐藏技术的方法，并对端口数据流采用基于规则检测的识别控制方式。具体的实现是：将通信主机双方隐藏在 Internet 之后，使它们不直接与 Internet 相连接。因此，它们之间的通信需要通过称为协议转换的代理程序来完成，相应的网络拓扑图如图 2。这样做的代价是，需要在服务端和客户端所在网络内都增加代理程序。由代理程序完成通信连接、数据流识别与控制、事件监听及数据转发功能。

图 2 主机隐藏

首先，定义代理程序之间的 UDP 数据包格式如下：

Datagram_for_proxy=MachineID+OriginalData (1)

其中，MachineID 表示主机标识，它是在代理程序之间建立连接之后交换的，OriginalData 表示原数据包，即 Xserver 或 Xclient 产生的初始数据包。这里的“+”表示 MachineID 是 OriginalData 的前缀。

数据流识别与控制是代理程序的主要功能之一，主要功能是防止 6000 或 177 端口被非法连接。这里采用基于规则的二次过滤算法来实现，算法描述如下。

算法 1 二次过滤算法

[输入]:代理程序的监听端口 xport 上接收到的数据流 x_in_data，UDP 检查次数 UDPTimes（第一次调用时的值为 0），连接方的 IP 地址 SourceIP

[处理]:

(1)如果 xport 是 XDMCP 端口

1)按照式(1)将主机标识 MachineID 及原数据包 OriginalData 从 x_in_data 中分离出来；

2)MachineID 是否符合代理程序指定的编码规则；

如果不是则转(3)

如果是，置 UDPTimes=UDPTimes+1，并作如下判断

如果 UDPTimes = 5 则表示 UDP 通信检查都通过，将 SourceIP 写入到一个一维数组 SourceRequestIP；转(3)；

(2)如果是 X-window 端口（6000）

检查数组 SourceRequestIP 中是否包含有 SourceIP 的记录；

如果是，则允许在 6000 端口上进行通信，并将 x_in_data 转发给 Xserver

否则，不将 x_in_data 转发给 Xserver，拒绝建立连接。

(3)处理结束

[输出]: UDPTimes；

是否允许下一次通信；

执行结果的成功与否。

3.2 通信加密

增加代理程序的另一个优点是，在不修改通信双方应用软件的情况可以实时地获得通信的数据包，因而可以在发送之前对这些数据进行加密处理。这样就可以解决 X-window 应用程序的数据传输安全问题。

目前通用的加密算法主要有以 DES 为代表对称密钥方法和以 RSA 为代表非对称公开密钥方法。以 C/S 为基础的互联网通信一般是要以非对称公开密钥的加密算法来保证通信的绝对安全性，但是鉴于不同的计算机终端平台，RSA 加密体制目前还不能适应于 X-window 的实时通信。为此本系统采用我们自主提出的基于神经网络混沌吸引子的非对称加密算法[5]。通信过程中的公开密钥是由代理程序在通信建立之前的协商阶段实现交换并确定。

因为，不同的 Xclient 对传输数据有不同程度的加密处理，而在 X-window 通信过程中的第一、二阶段，是双方参数协商的阶段，传输着一些敏感信息。因此，目前我们所开发的系统只对这部分传输数据进行加密。

3.3 二次身份认证

采用了主机隐藏技术后，客户端就无法看到远程主机，因而不能直接访问它。因此必须通过代理程序登录到远程主机，再进入 X-window 应用系统。参考图 2 的网络结构，具体的处理流程描述如下：

（1）代理程序 2 连接到代理程序 1 的指定端口；

（2）代理程序 2 发送命令“Connect”，其参数为用户名 username、密码 password；

（3）代理程序 1 获得 username 及 password，并到事先设定的数据库中对用户信息进行认证；

（4）如果不能通过用户认证，代理程序 1 发送认证错误信息给代理程序 2，并转步骤（6），否则，继续下一步执行；

（5）客户端软件启动后，即开始进行 X-window 的 4 次通信过程，包括 X-window 本身所要求的身份认证。处理完后，结束流程；

（6）代理程序关闭本地监听端口，并显示出错信息，结束流程；

这里的二次身份认证是在代理程序这个层次上完成的，对具体的 X-window 应用来说是透明的，也就是说，X-window 应用不需要任何修改就可以实现二次身份认证。而且整个系统的安全等级就比单纯的 X-window 认证提高了。

(下转第 29 页)

而处于挂起状态，亦无法判断是否存在处于 Grant 状态的盟员。

以上所有的结论均是指出了死锁产生的必要条件。下面的定理给出死锁产生的一个充分条件。

定理 3 采用 Frederick 算法的联邦，所有盟员间既相互控制又相互受限。在联邦运行过程中，如果进行到下面的一种状态：(1) 存在至少两个因 NMR 或 NMRA 请求而被挂起的盟员，满足 T＞GALT，且 LETS＞GALT；(2) 在满足上面要求的盟员中，存在两个盟员，其输出时间是所有盟员中最小的，并且彼此相等；(3) 上述两个盟员的输出时间与它们的 GALT 相等。则该联邦运行将处于死锁状态。

证明 假设盟员 k 和盟员 j 处于上述的状态之中，它们的输出时间最小，彼此相等，且等于它们的 GALT。对盟员 k 而言，它不会收到任何小于 GALT(k)的消息，因此，在以后的处于挂起状态的时间里，在它的消息队列中不会收到 LETS(k)＜GALT(k)的消息。因此，由 Frederick 算法的式（2）可知，在以后的处于挂起状态的时间里，盟员 k 的输出时间将取决于它自己的 GALT(k)。由 Frederick 算法的式（1）可知，GALT(k)则取决于具有最小输出时间的盟员 j。同样的道理，盟员 j 的输出时间也将取决于它自己的 GALT(j)，而 GALT(j)则取决于具有最小输出时间的盟员 k。盟员 k 和盟员 j 之间互相依赖。盟员 k 解除挂起状态的关键在于 GALT(k)的增加，而 GALT(k)的增加，则依赖于盟员 j 的输出时间的增加，盟员 j 的输出时间的增加，则依赖于 GALT(j)的增加，GALT(j)的增加，又依赖于盟员 k 的输出时间的增加，最后，盟员 k 的输出时间的增加，反过来依赖于 GALT(k)的增加。于是，GALT(k)的增加就陷入了死锁状态。GALT(j)也将如此。其他盟员的 GALT 也将保持不变，其逻辑时间将无法推进超过 GALT，最终将全部处于挂起状态而陷入死锁。证毕。

由定理 3 可以看出，局部的死锁可以导致整个联邦运行都处于时间无法向前推进的死锁状态。

由以上的分析可知，死锁产生的原因不是 HLA 规范本身所造成的，其根本原因在于不恰当的 GALT 算法。由于 HLA 主要面向大规模的系统仿真，并且参加仿真的盟员可以使用多种时间推进机制，因此死锁问题一般是不会出现的。但是对于一个合格的 RTI 产品而言，必须综合考虑各种情况，尤其是死锁问题。

4 结束语

GALT 算法是时间管理服务能否实现的关键技术，不合理的 GALT 算法会导致死锁，从而导致整个仿真无法向前推进。Frederick 算法在特殊情况下会造成死锁。因此，在设计时间推进时，若采用的是该算法，在联邦设计时应避免出现类似定理 3 所描述的这种特殊情况。若设计新的 GALT 算法，则在避免死锁的同时，必须确保符合 HLA 时间管理的原则。

参考文献

1 Module1. Basic Concepts of the High Level Architecture (HLA) [EB/OL]. Mcleod Institute of Simulation Science. http://www.ecst.csuchico.edu/~mcleod, 1998

2 IEEE Std 1516.1-2000. IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) —— Federate Interface Specification. 2000

3 欧阳伶俐, 宋 星, 卿杜政等. HLA 时间管理与 PDES 仿真算法研究[J]. 系统仿真学报, 2000, 12(3): 237-240

4 Kuhl F, Weatherly R, Dahmann J. 计算机仿真中的 HLA 技术[M]. 北京: 国防工业出版社，2003-06

5 刘步权, 王怀民, 姚益平. 一种无死锁的时间管理算法[J]. 软件学报, 2003, 14(9):1515-1522

(上接第 23 页)

3.4 实现方法

X-window 应用软件的运行环境一般在工作站上，是基于 Unix 系统，而客户端可能在 Windows 或 Liunx 等操作系统上运行。因此，在实现上采用基于 Java 虚拟机（Java Virtual Machine,JVM），并用 Java 语言的面向对象方法实现，保证代理程序具有足够的可移植性。

在实现中，将相关的设计功能独立成类，类之间的调用通过 public 属性或方法。同时充分利用 Java 语言对线程的支持，在处理实时数据时，采用了线程的方式。相关的类有：

（1）MainClass。这是代理程序的主类，负责用户界面、DES 密钥的生成协商以及 TCPMain、UDPMain、Authentic 类的调用。

（2）TCPMain。负责在本地启动 X-window 端口 6000 监听，接受连接请求，调用 RuleCheck 对请求数据包进行规则检查。并针对不同的连接调用读写线程 ReadWrite。

（3）UDPMain。负责在本地启动 XDMCP 端口 177 的 UDP 类型监听，接受连接请求，调用 RuleCheck 对请求数据包进行规则检查。采用单线程的方式；并针对 X-window 的 5 次 UDP 通信握手中的数据包调用 CNNProcess 进行数据的加密、解密处理。

（4）ReadWrite。为某个 TCP 连接建立单独的收发线程，以便实时地为每个连接转发数据包，最大限度地减少引入代理程序而带来的传输延迟。

（5）CNNProcess。实现 UDP 数据包进行基于神经网络混沌吸引子的非对称加密、解密处理。

（6）RuleCheck。进行规则检查。

（7）Authentic。进行用户的身份认证，需要从本地的一个持久存储中读取用户信息，并进行比较。

4 总结

X-window 应用程序采用基于 C/S 的软件结构，但它要求通信双方都需要打开监听端口。如果在 Internet 环境中应用这种结构，必然会产生很多的安全问题，主要的根本原因是通信双方主机暴露于 Internet 而导致端口非法连接产生的问题，以及通信过程的信息安全和 X-window 过于简单的用户认证带来的安全隐患。本文针对这些问题，提出了一种新型实用的安全保证方案，从主机隐藏、通信加密及二次身份认证等 3 个方面构造了一个安全体系，使普通的 X-window 应用程序的安全等级大大提高了。而且根据这个方案，不需要修改已有的 X-window 应用软件就可以解决安全问题。最后，给出了这种安全体系结构的基于 JVM 实现方法，并直接应用到我们所开发的 Web-EDA 技术平台中。从 Web-EDA 平台的运行情况上看，本方案是可靠的。

参考文献

1 唐多元, 宋艺新, 于庆来等. X 窗口系统在电站仿真机中的应用. 清华大学学报(自然科学版), 1998,(5)

2 Ye A N. Xlib Programming Manual. USA: O'Reilly & Associates, Inc., 1990

3 崔 霖, 蔡亦波, 施亦欣. X-Window 的安全问题. 计算机工程, 2000, 36(6): 53-55

4 Zeng Jianping, Guo Donghui. A Prototype of Web-based Middleware System for EDA Tools Sharing. Proceedings of the 8th International Conference on Computer Supported Cooperative Work in Design, 2004-05

5 刘年生. 神经网络混沌加密算法及其在下一代互联网安全通信中的应用研究[博士学位论文]. 厦门大学, 2003-10